Try-Mopsa
Relational Static Analysis in Your Pocket

Raphaël Monat

rmonat.fr

Dagstuhl #25241

October 2025

Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- Allows users to choose their composition of abstract domains

Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- Allows users to choose their composition of abstract domains
- Eases development of highly expressive relational static analyses

Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- Allows users to choose their composition of abstract domains
- Eases development of highly expressive relational static analyses
- Can be used as an experimentation platform

Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

- Allows users to choose their composition of abstract domains
- Eases development of highly expressive relational static analyses
- Can be used as an experimentation platform
  - Supports large subsets of C, Python

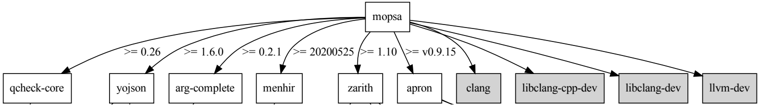
Modular **O**pen **P**latform for **S**tatic **A**nalysis [Jou+19]

`gitlab.com/mopsa/mopsa-analyzer` or `opam install mopsa`

Started by ERC Consolidator Grant (2016-2021) of Antoine Miné (LIP6, SU)

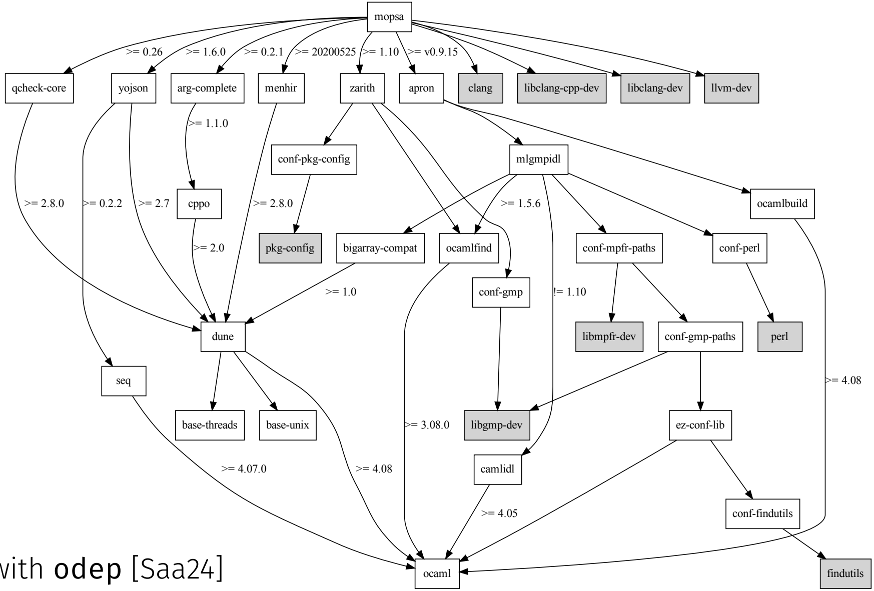
- Allows users to choose their composition of abstract domains
- Eases development of highly expressive relational static analyses
- Can be used as an experimentation platform
  - Supports large subsets of C, Python
  - Academically competitive on real-world benchmarks (SV-Comp)

Static analyzers are complex pieces of software!



Generated with **odep** [Saa24]

Static analyzers are complex pieces of software!



Generated with **odep** [Saa24]

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

Try-Mopsa

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- Zero-install

Try-Mopsa

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- Zero-install
- Supports core features of Mopsa

Try-Mopsa

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- Zero-install
- Supports core features of Mopsa
- Features responsive interface supporting smartphones to computers

Try-Mopsa

Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- Zero-install
- Supports core features of Mopsa
- Features responsive interface supporting smartphones to computers
- Scales in number of users (purely client-side)

Try-Mopsa

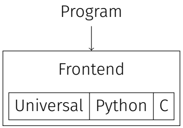
Try-Mopsa: a scaled down version of Mopsa running purely in browsers.

- Zero-install
- Supports core features of Mopsa
- Features responsive interface supporting smartphones to computers
- Scales in number of users (purely client-side)
- Can be convenient for onboarding or teaching?

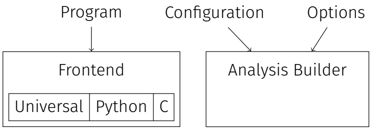
Outline

Under the Hood of Try-Mopsa

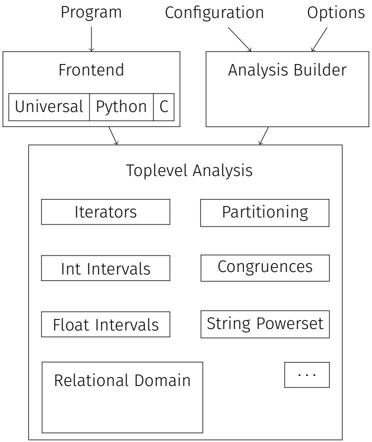
Overview of Mopsa



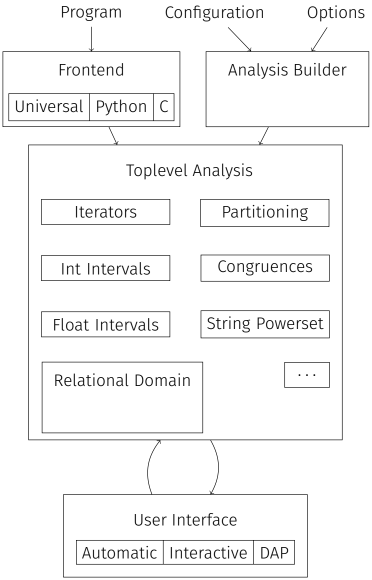
Overview of Mopsa



Overview of Mopsa



Overview of Mopsa

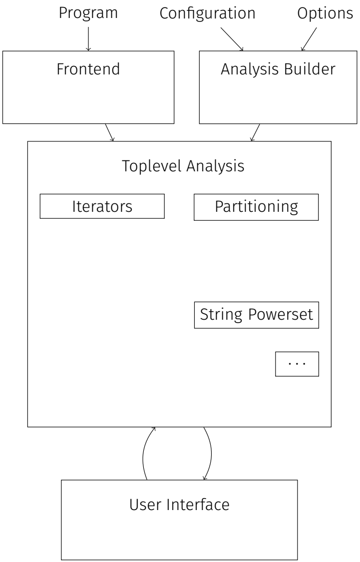


Compiling Mopsa to JavaScript

Rely on Js_of_ocaml [VB14]

89% of Mopsa's codebase is written in OCaml

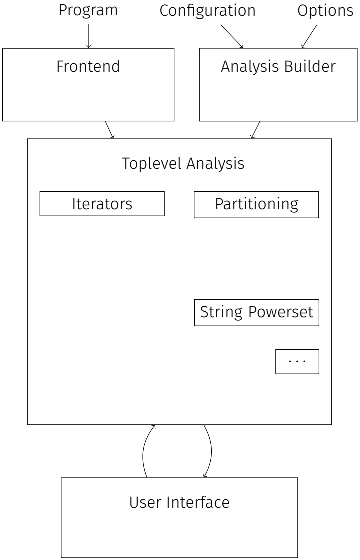
Compiling Mopsa to JavaScript



Rely on Js_of_ocaml [VB14]

89% of Mopsa’s codebase is written in OCaml

Compiling Mopsa to JavaScript



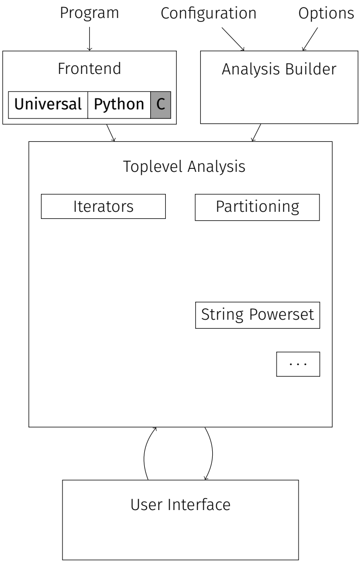
Rely on Js_of_ocaml [VB14]

Parsing libraries

Menhir

~~libclang~~: too much manual work for now

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

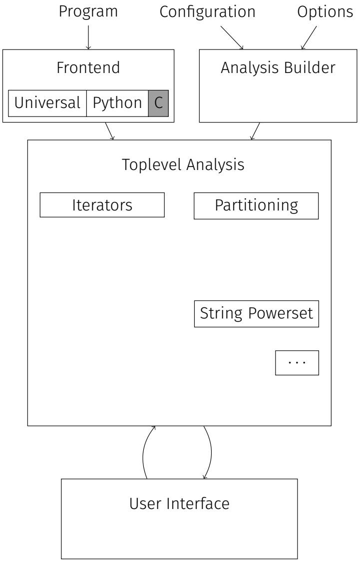
Rely on Js_of_ocaml [VB14]

Parsing libraries

Menhir

~~libclang~~: too much manual work for now

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

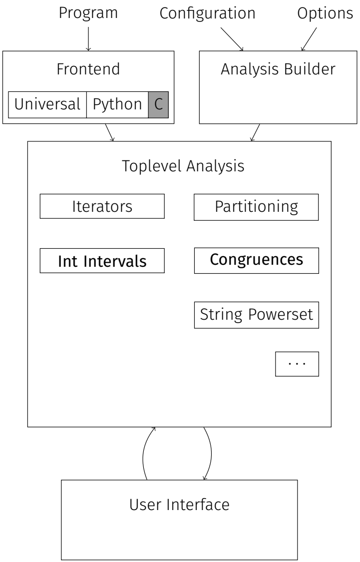
Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

~~`Zarith`~~ ⇝ `Zarith_stubs_js`

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

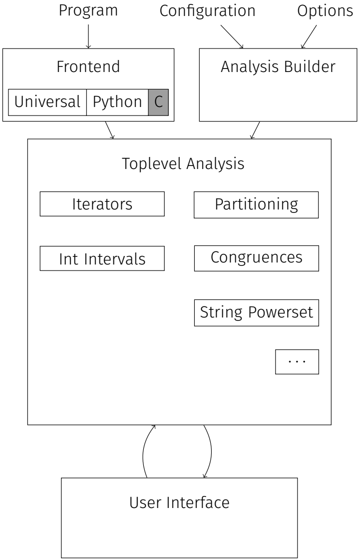
Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

~~Zarith~~ ⇝ Zarith_stubs_js

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

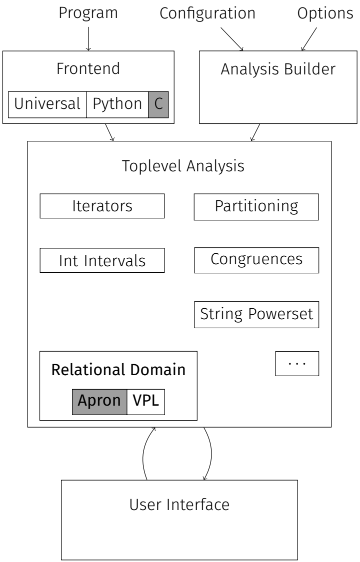
Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

~~Apron [JM09]~~ $\rightsquigarrow$ VPL [Bou+18] w/ extensions

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

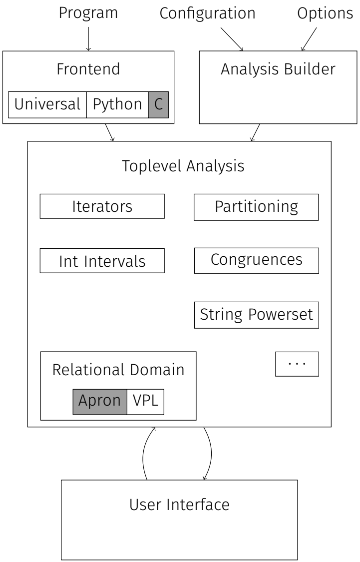
Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

~~Apron [JM09]~~ $\rightsquigarrow$ VPL [Bou+18] w/ extensions

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

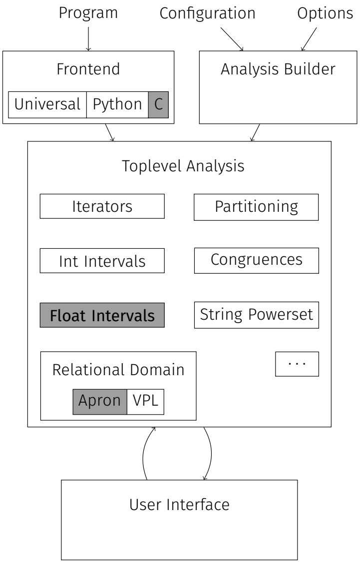
Relational domains

~~Floating-point~~

Our implementation needs to set rounding modes

Not supported in JavaScript/WebAssembly.

Compiling Mopsa to JavaScript



Gray = unsupported by Try-Mopsa.

Rely on Js_of_ocaml [VB14]

Parsing libraries

Arbitrary-precision integer arithmetic

Relational domains

~~Floating-point~~

Our implementation needs to set rounding modes

Not supported in JavaScript/WebAssembly.

Toplevel

- Handles user actions

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor   with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel
- Relies on virtual filesystem to embed stubs and configurations

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel
- Relies on virtual filesystem to embed stubs and configurations

Handling User Inputs

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor   with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel
- Relies on virtual filesystem to embed stubs and configurations

Handling User Inputs

- Mopsa CLI: synchronous, blocking interaction loop

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel
- Relies on virtual filesystem to embed stubs and configurations

Handling User Inputs

- Mopsa CLI: synchronous, blocking interaction loop
- But web interfaces require asynchronous interfaces!

Handling Web Interactions

Toplevel

- Handles user actions
- Ace editor with custom highlighting and ANSI rendering
- Dynamic option generation
- Triggers/interrupts web worker

Web Worker

- Contains Mopsa in JS
- Intercepts `std{in,out}`, forwards them to the toplevel
- Relies on virtual filesystem to embed stubs and configurations

Handling User Inputs

- Mopsa CLI: synchronous, blocking interaction loop
- But web interfaces require asynchronous interfaces!

$\Longrightarrow$ Rely on `sync-message` [Moj23] (low-level synchronous comm.)

Interface Overview

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}

assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}

assert(0 <= j < |s|  ⟹  s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}

assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}

assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}

assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}
// s ∈ {"a", "ab", "abc", ..., "abc···xy", "abc···xyz"}
assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

Demo!

Running example

```
str s = "a";
int i = 1;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}
// s ∈ {"a", "ab", "abc", ..., "abc···xy", "abc···xyz"}
assert(0 <= j < |s| ⟹ s[j] - 'a' < |s|);
```

$\Longrightarrow$ `try-mopsa.rmonat.fr`

Implementation Discussion

Implementation

- Mopsa $\simeq 87k$LOC

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa
  - 1,500 lines of HTML/CSS/JS

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa
  - 1,500 lines of HTML/CSS/JS
  - 670 lines of OCaml (toplevel/worker)

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa
  - 1,500 lines of HTML/CSS/JS
  - 670 lines of OCaml (toplevel/worker)
  - 500 lines of OCaml (VPL)

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa
  - 1,500 lines of HTML/CSS/JS
  - 670 lines of OCaml (toplevel/worker)
  - 500 lines of OCaml (VPL)
  - Compiled, optimized size: 3.1 megabytes

Implementation

- Mopsa $\simeq 87k$LOC
- Try-Mopsa
  - 1,500 lines of HTML/CSS/JS
  - 670 lines of OCaml (toplevel/worker)
  - 500 lines of OCaml (VPL)
  - Compiled, optimized size: 3.1 megabytes
  - *No breaking changes*

Program	Binary execution	Firefox execution
attributes.py		
fspath		2
list.py		3
loop.p		3
recen		3
str_alphabet.u	0.04s ± 0.01	0.20s ± 0.01
str_alphabet2.u	0.23s ± 0.01	0.81s ± 0.01
str_conc_loop.u	0.08s ± 0.01	0.29s ± 0.01
str_conc_loop2.u	0.04s ± 0.01	0.15s ± 0.01

- 5× slowdown compared to native binary
- Raw performance is not a priority
- Still reasonable analysis times

Browser compatibility

Playwright framework to asses the browser compatibility of Try-Mopsa

Conformance tests

5 tests x (3 desktop browsers + 2 mobiles browsers x 2 viewports)

Identified several rendering issues on mobile

Browser compatibility

Playwright framework to asses the browser compatibility of Try-Mopsa

Conformance tests

5 tests x (3 desktop browsers + 2 mobiles browsers x 2 viewports)

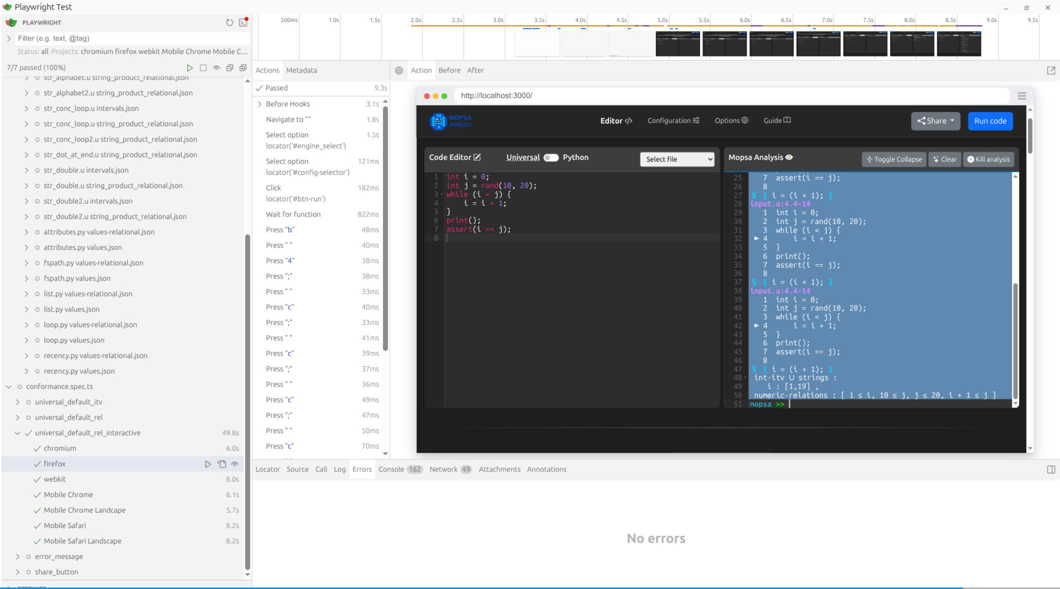
Identified several rendering issues on mobile

Additional tests

Universal: 13 programs x 11 configurations

Python: 5 programs x 6 configurations

Browser compatibility (II)



Conclusion

Related Work

- Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
  Less adaptation required, but maintenance and security issues

Related Work

- Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
  Less adaptation required, but maintenance and security issues
- Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25] Binsec tutorial [RB25]
  Try-Mopsa supports relational domains and interactive user input

Related Work

- Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
  Less adaptation required, but maintenance and security issues
- Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25] Binsec tutorial [RB25]
  Try-Mopsa supports relational domains and interactive user input
- LiSA for teaching [Neg+24]

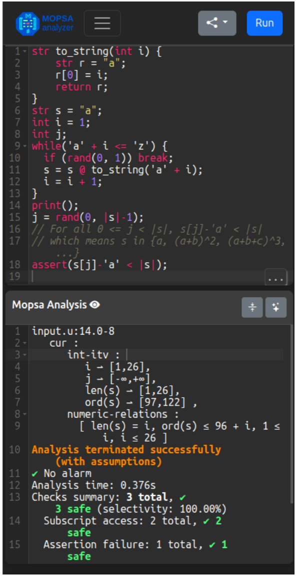
Related Work

- Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
  Less adaptation required, but maintenance and security issues
- Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25] Binsec tutorial [RB25]
  Try-Mopsa supports relational domains and interactive user input
- LiSA for teaching [Neg+24]
- Interaction with IDEs [LDB19]

Related Work

- Server-side: Interproc [Jea09], Banal [Min12], FuncTion [Urb15]
  Less adaptation required, but maintenance and security issues
- Client-side: Ciao Prolog [MPH22], Salto (OCaml) [LM25] Binsec tutorial [RB25]
  Try-Mopsa supports relational domains and interactive user input
- LiSA for teaching [Neg+24]
- Interaction with IDEs [LDB19]
- Works between concrete and abstract debuggers [Do+20; Hol+24; MVR23]

Conclusion



- Pure JS version of Mopsa

Conclusion

MOPSA analyzer

Run

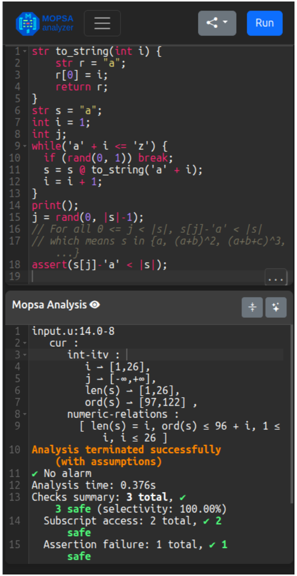
```
str to_string(int i) {
    str r = "a";
    r[0] = i;
    return r;
}
str s = "a";
int i = 1;
int j;
while('a' + i <= 'z') {
  if (rand(0, 1)) break;
  s = s @ to_string('a' + i);
  i = i + 1;
}
print();
j = rand(0, |s|-1);
// For all 0 <= j < |s|, s[j]-'a' < |s|
// which means s in {a, (a+b)^2, (a+b+c)^3, ...}
assert(s[j]-'a' < |s|);
```

Mopsa Analysis

```
input.u:14.0-8
  cur :
    int-itv :
      i ↦ [1,26],
      j ↦ [-∞,+∞],
      len(s) ↦ [1,26],
      ord(s) ↦ [97,122] ,
    numeric-relations :
      [ len(s) = i, ord(s) ≤ 96 + i, 1 ≤ i, i ≤ 26 ]
Analysis terminated successfully (with assumptions)
✔ No alarm
Analysis time: 0.376s
Checks summary: 3 total, ✔ 3 safe (selectivity: 100.00%)
  Subscript access: 2 total, ✔ 2 safe
  Assertion failure: 1 total, ✔ 1 safe
```

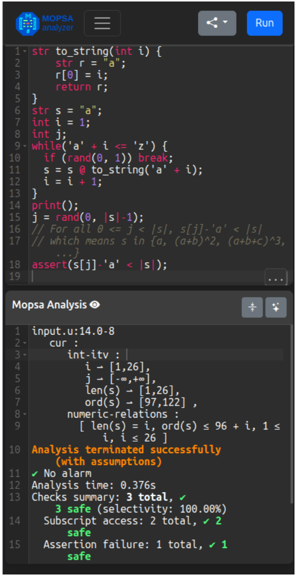
- Pure JS version of Mopsa
- Supports polyhedra, abstract debugguer

Conclusion



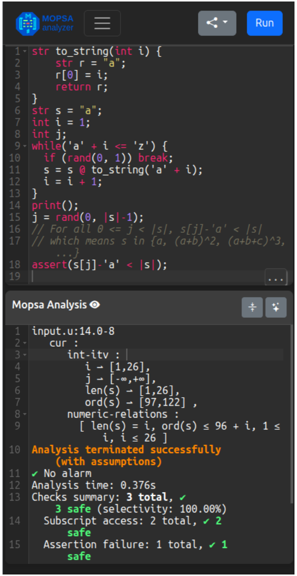
- Pure JS version of Mopsa
- Supports polyhedra, abstract debugguer
- Works on smartphones and desktops alike

Conclusion



- Pure JS version of Mopsa
- Supports polyhedra, abstract debugguer
- Works on smartphones and desktops alike
- Useful for demo or lightweight teaching purposes

Conclusion



- Pure JS version of Mopsa
- Supports polyhedra, abstract debugguer
- Works on smartphones and desktops alike
- Useful for demo or lightweight teaching purposes

`try-mopsa.rmonat.fr`

References – I

[Bou+18] Sylvain Boulmé et al. **“The Verified Polyhedron Library: an Overview”.** In: IEEE, 2018, pp. 9–17. DOI: `10.1109/SYNASC.2018.00014`.

[Do+20] Lisa Nguyen Quang Do et al. **“Debugging Static Analysis”.** In: IEEE Trans. Software Eng. 7 (2020), pp. 697–709. DOI: `10.1109/TSE.2018.2868349`.

[Hol+24] Karoliine Holter et al. **“Abstract Debuggers: Exploring Program Behaviors using Static Analysis Results”.** In: ed. by Jonathan Edwards and Marcel Taeumel. ACM, 2024, pp. 130–146. DOI: `10.1145/3689492.3690053`.

References – II

[Jea09] Bertrand Jeannet. Web Interface for the Interproc Analyzer. 2009. URL: https://pop-art.inrialpes.fr/interproc/interprocweb.cgi.html.

[JM09] Bertrand Jeannet and Antoine Miné. “Apron: A Library of Numerical Abstract Domains for Static Analysis”. In: Springer, 2009, pp. 661–667.

[Jou+19] M. Journault et al. “Combinations of reusable abstract domains for a multilingual static analyzer”. In: New York, USA, July 2019, pp. 1–17.

References – III

[LDB19] Linghui Luo, Julian Dolby, and Eric Bodden. "MagpieBridge: A General Approach to Integrating Static Analyses into IDEs and Editors (Tool Insights Paper)". In: ed. by Alastair F. Donaldson. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 21:1–21:25. DOI: 10.4230/LIPICS.ECOOP.2019.21.

[LM25] Pierre Lermusiaux and Benoît Montagu. Web Demo for the Salto Analyzer. 2025. URL: https://salto.gitlabpages.inria.fr/demo/salto.html.

[Min12] Antoine Miné. Web Interface for Sufficient Condition Polyhedral Prototype Analyzer. 2012. URL: https://mine.perso.lip6.fr/banal/.

References – IV

[Moj23] Alex Mojaki. sync-message. Version 0.0.12. Oct. 2023.

[MPH22] Jose F. Morales, Guillermo García Pradales, and Manuel Hermenegildo. Ciao Prolog Playgrund. 2022. URL: https://ciao-lang.org/playground/.

[MVR23] Mats Van Molle, Bram Vandenbogaerde, and Coen De Roover. “Cross-Level Debugging for Static Analysers”. In: ed. by João Saraiva, Thomas Degueule, and Elizabeth Scott. ACM, 2023, pp. 138–148. DOI: 10.1145/3623476.3623512.

References – V

[Neg+24] Luca Negrini et al. **"Teaching Through Practice: Advanced Static Analysis with LiSA".** In: ed. by Emil Sekerinski and Leila Ribeiro. Lecture Notes in Computer Science. Springer, 2024, pp. 43–57. DOI: `10.1007/978-3-031-71379-8_3`.

[RB25] Frédéric Recoules and Sébastien Bardin. BINSEC Tutorial at PLDI'25: Adapting Symbolic Execution for Binary-level Sec 2025. URL: `https://binsec.github.io/tutorial-pldi2025/`.

[Saa24] Simmo Saan. Odep: Dependency graphs for OCaml modules, libraries and packages. Version 0.2.1. Apr. 2024.

References – VI

[Urb15] Caterina Urban. Web Interface for FuncTion. 2015. URL: https://www.di.ens.fr/~urban/FuncTion.html.

[VB14] Jérôme Vouillon and Vincent Balat. "From bytecode to JavaScript: the Js_of_ocaml compiler". In: Softw. Pract. Exp. 8 (2014), pp. 951–972. DOI: 10.1002/SPE.2187.